Customer Segmentation Analysis - K-Means Clustering and RFM Analysis

February 9, 2024

0.1 Project: Customer Segmentation and Analysis

0.1.1 Stage Three: RFM (Recency, Frequency, Monetary) Analysis

0.1.2 Introduction:

In this stage of the customer segmentation analysis, I delved into the dataset(cleaned) of the online retail store to unravel the intricacies of customer behaviour. By leveraging the powerful RFM (Recency, Frequency, Monetary) analysis and K-Means Clustering algorithm, I aimed to distill valuable insights that could profoundly impact business strategies and enhance customer-centric decision-making.

0.1.3 Dataset Overview:

The dataset in this stage of the project encompasses a diverse array of almost 400K transactions, capturing the interactions of customers on the online retail platform. Features such as transaction dates(InvoiceDate), purchase amounts(UnitPrice), product quantity, stock code and customer identifiers form the foundation for a comprehensive exploration of customer dynamics.

0.1.4 Objectives of the Analysis:

The primary Objectives of this analysis are as follows:

1. Segmentation for Potential Targeted Marketing:
    - How can customers be categorized into distinct segments based on thier recency, frequency, and monetary contributions?
    - What insights can these segments provide to guide marketing strategies for improved customer engagements?
2. Identifying High-Value Customers:
    - Can we identify or pinpoint high-value customers who contribute significantly to the business's revenue?
    - What patterns in recency, frequency, and monetary metrics characterize these high-value customers?
3. Understanding the correlation between RFM metrics:
    - How does recency and purchasing frequency affect monetary contributions of each customer segment?
4. Customizing Communication Strategies:
    - How can communication strategies be customized for different customer segments to enhance the overall customer experience

- What personalized approaches can be adopted based on the identified RFM segments?

By delving into the recency, frequency and monetary dimensions of customer transactions, this analysis aims to provide actionable insights that empower the online retail store to optimize marketing efforts, improve customer engagement and experience.

[1]:
```python
import pandas as pd
import numpy as np
import plotly.express as px
import matplotlib.pyplot  as plt
%matplotlib inline
```

[2]:
```python
df = pd.read_csv('online_retail_cl.csv')
df.head()
```

[2]:
```
   InvoiceNo  StockCode                          Description  Quantity  \
0     536365      85123   WHITE HANGING HEART T-LIGHT HOLDER         6
1     536365      71053                  WHITE METAL LANTERN         6
2     536365      84406       CREAM CUPID HEARTS COAT HANGER         8
3     536365      84029  KNITTED UNION FLAG HOT WATER BOTTLE         6
4     536365      84029       RED WOOLLY HOTTIE WHITE HEART.         6

          InvoiceDate  UnitPrice  CustomerID         Country TransactionType  \
0  2010-12-01 08:26:00       2.55       17850  United Kingdom            sale
1  2010-12-01 08:26:00       3.39       17850  United Kingdom            sale
2  2010-12-01 08:26:00       2.75       17850  United Kingdom            sale
3  2010-12-01 08:26:00       3.39       17850  United Kingdom            sale
4  2010-12-01 08:26:00       3.39       17850  United Kingdom            sale

   TotalAmount MonthWeek MonthofYear
0        15.30        W1         Dec
1        20.34        W1         Dec
2        22.00        W1         Dec
3        20.34        W1         Dec
4        20.34        W1         Dec
```

[3]:
```python
df.shape
```

[3]: (399654, 12)

[4]:
```python
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 399654 entries, 0 to 399653
Data columns (total 12 columns):
 #   Column           Non-Null Count   Dtype
---  ------           --------------   -----
 0   InvoiceNo        399654 non-null  int64
 1   StockCode        399654 non-null  int64
```

```
 2   Description      399654 non-null  object
 3   Quantity         399654 non-null  int64
 4   InvoiceDate      399654 non-null  object
 5   UnitPrice        399654 non-null  float64
 6   CustomerID       399654 non-null  int64
 7   Country          399654 non-null  object
 8   TransactionType  399654 non-null  object
 9   TotalAmount      399654 non-null  float64
 10  MonthWeek        399654 non-null  object
 11  MonthofYear      399654 non-null  object
dtypes: float64(2), int64(4), object(6)
memory usage: 36.6+ MB
```

0.1.5 1. Converting InvoiceDate from object to datetime datatype

[5]:
```
df['InvoiceDate'] = pd.to_datetime(df['InvoiceDate'])
```

[6]:
```
df.dtypes
```

[6]:
```
InvoiceNo                   int64
StockCode                   int64
Description                object
Quantity                    int64
InvoiceDate        datetime64[ns]
UnitPrice                 float64
CustomerID                  int64
Country                    object
TransactionType            object
TotalAmount               float64
MonthWeek                  object
MonthofYear                object
dtype: object
```

[ ]:
```

```

0.1.6 2. Calculate RFM (Recency, Frequency, Monetary) Metrics

2.1 Calculating Recency

[7]:
```
# Getting the most recent date from our dataset
most_recent_date = df['InvoiceDate'].max()
print('Most recent date:', most_recent_date)
```

```
Most recent date: 2011-12-09 12:50:00
```

[8]:
```
recency = most_recent_date - df.groupby('CustomerID')['InvoiceDate'].max()
recency
```

[8]:
```
CustomerID
12346   325 days 02:33:00
12347     1 days 20:58:00
12348    74 days 23:37:00
12349    18 days 02:59:00
12350   309 days 20:49:00
               …
18280   277 days 02:58:00
18281   180 days 01:57:00
18282     7 days 01:07:00
18283     3 days 00:48:00
18287    42 days 03:21:00
Name: InvoiceDate, Length: 4363, dtype: timedelta64[ns]
```

2.2 Calculating Frequency

[9]:
```
frequency = df.groupby('CustomerID')['InvoiceNo'].nunique()
frequency
```

[9]:
```
CustomerID
12346     2
12347     7
12348     4
12349     1
12350     1
         ..
18280     1
18281     1
18282     3
18283    16
18287     3
Name: InvoiceNo, Length: 4363, dtype: int64
```

2.3 Calculating Monetary

[10]:
```
monetary = df.groupby('CustomerID')['TotalAmount'].sum()
monetary
```

[10]:
```
CustomerID
12346    154367.20
12347      4310.00
12348      1437.24
12349      1457.55
12350       294.40
           …
18280       180.60
18281        80.82
18282       179.50
18283      2039.58
```

```
18287      1837.28
Name: TotalAmount, Length: 4363, dtype: float64
```

0.1.7 3. Create RFM Table

3.1 Combine all RFM Metrics into a single table

[11]:
```
merge_rfm = pd.merge(recency.dt.days, frequency, on ='CustomerID')
rfm_table = pd.merge(merge_rfm, monetary, on='CustomerID')
```

[12]:
```
rfm_table = rfm_table.rename(columns={
    'InvoiceDate': 'Recency',
    'InvoiceNo' : 'Frequency',
    'TotalAmount' : 'Monetary'
})
rfm_table
```

[12]:
```
            Recency  Frequency   Monetary
CustomerID
12346           325          2  154367.20
12347             1          7    4310.00
12348            74          4    1437.24
12349            18          1    1457.55
12350           309          1     294.40
…             …          …        …
18280           277          1     180.60
18281           180          1      80.82
18282             7          3     179.50
18283             3         16    2039.58
18287            42          3    1837.28

[4363 rows x 3 columns]
```

[13]:
```
# Checking the dimension of our generated RFM table
rfm_table.shape
```

[13]:
```
(4363, 3)
```

The shape of the RFM table indicates that it has a total of **4363** rows and **3** columns

0.1.8 4. Performing K-Means Clustering on the dataset

The K-Means clustering method is an unsupervised machine learning technique or algorithm used to identify and group clusters of data objects in a dataset based on similarities.

4.1 Standardize The Data Standardizing the RFM values to ensure equal weight during clustering

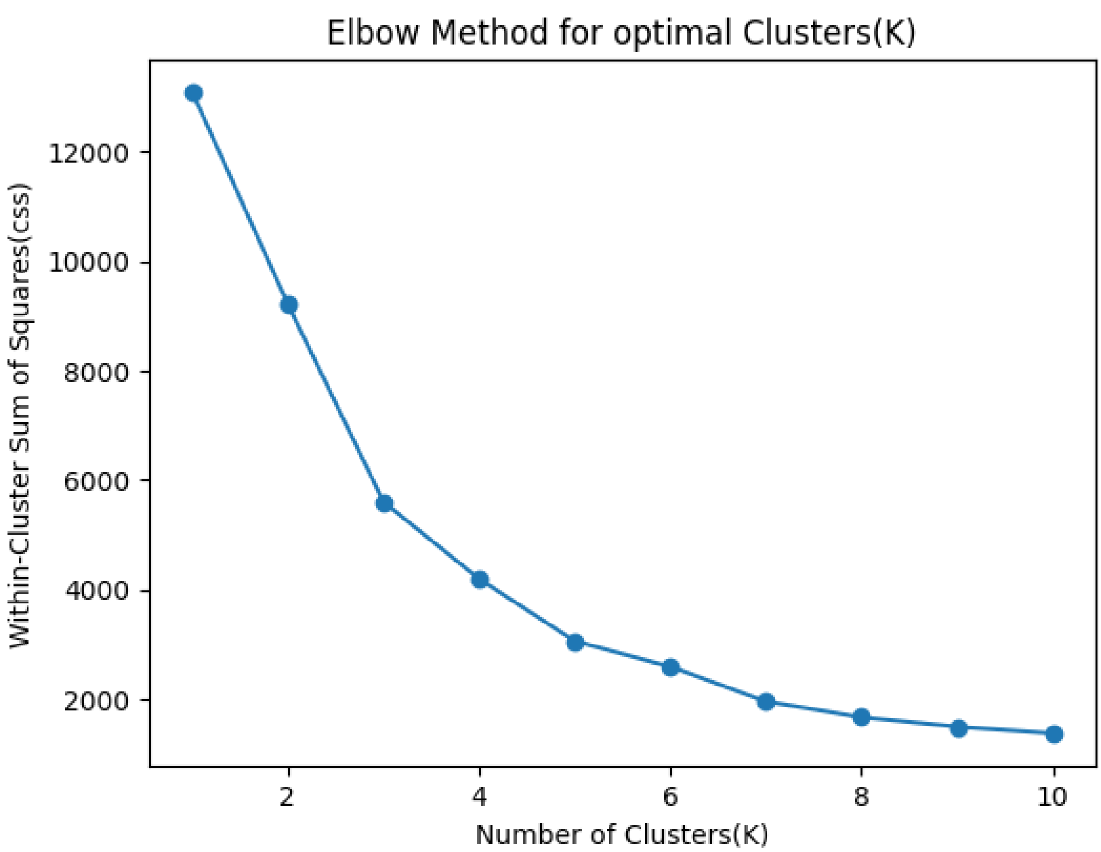
```
[14]: from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
rfm_scaled = scaler.fit_transform(rfm_table)
```

4.2 Choosing Number of Cluster(K)

```
[15]: from sklearn.cluster import KMeans

css = []
for i in range(1,11):
    kmeans = KMeans(n_clusters=i, n_init = 'auto' , init='k-means++',␣
 ↪random_state=42)
    kmeans.fit(rfm_scaled)
    css.append(kmeans.inertia_)
```

4.3 Plotting the Elbow Method

```
[16]: plt.plot(range(1,11), css, marker = 'o')
plt.title('Elbow Method for optimal Clusters(K)')
plt.xlabel('Number of Clusters(K)')
plt.ylabel('Within-Cluster Sum of Squares(css)')
plt.show()
```



From the elbow method plot above, the number of clusters in our dataset will be 3.

4.4 Applying K-Means Clustering

[17]:
```
# Based on the Elbow Method our cluster(K) is 3
kmeans = KMeans(n_clusters = 3, n_init = 10, init = 'k-means++', random_state =␣
↪42)
rfm_table['Cluster'] = kmeans.fit_predict(rfm_scaled)+1
rfm_table
```

[17]:
```
            Recency  Frequency   Monetary  Cluster
CustomerID
12346           325          2  154367.20        3
12347             1          7    4310.00        1
12348            74          4    1437.24        1
12349            18          1    1457.55        1
12350           309          1     294.40        2
…               …          …        …        …
18280           277          1     180.60        2
18281           180          1      80.82        2
18282             7          3     179.50        1
18283             3         16    2039.58        1
18287            42          3    1837.28        1

[4363 rows x 4 columns]
```

4.5 Grouping Each Clusters by the Means of Recency, Frequency and Monetary

[18]:
```
# Exploring the clusters
cluster_means = rfm_table.groupby('Cluster').mean()
cluster_sizes = rfm_table.groupby('Cluster').value_counts()

print(cluster_means)
```

```
            Recency  Frequency       Monetary
Cluster
1         38.784810    5.65761    2005.423939
2        245.192238    1.83213     577.414116
3         25.687500   89.56250  129591.279375
```

From the above means generated with respect to each cluster, we notice that: 1. In terms of **'Recency'**, `Cluster 3` had the least value which meant customers from that cluster performed a transaction or visited the site most recently. This is followed by `Cluster 1` with the next least recency value. `Cluster 2` recorded the highest recency value, meaning it has been a long time since customers of that cluster performed any transaction on or visited the online platform.

2. In terms of how **'Frequent'** customers in each cluster transacted on the platform, `Cluster 3` recorded the highest frequency of the three clusters, followed by customers in `Cluster 1`,

with customers belonging to `Cluster 2` recording the least or lowest mean frequency of the three cluster groups.

3. For **'Monetary'**, the same observation can be made. `Cluster 3` recorded the highest mean monetary value among the three clusters. This is followed by `Cluster 1`, with `Cluster 2` recording the lowest mean monetary value of the three cluster groups.

[19]:
```
print(cluster_sizes)
```

```
Cluster  Recency  Frequency  Monetary
1        0        1          227.39       1
         91       2          547.90       1
                  6          765.70       1
                  4          1007.03      1
                  3          1255.00      1
                                         ..
3        7        49         199206.05    1
         9        64         119434.39    1
         23       24         125490.88    1
         38       66         85046.25     1
         325      2          154367.20    1
Name: count, Length: 4363, dtype: int64
```

4.6 Implementing Customer Categorization Rules Based on Cluster Means For this, three levels of categorization will be used to represent the three customer clusters identified.

1. High-Value: These customers exhibit the best RFM values and are the most valuable to the business. Criteria include;
   - Lower recency
   - High frequency
   - High monetary
2. Average: This group of customers have moderate RFM values. Criteria for this category include:
   - Moderate recency (not the most recent but not too old)
   - Moderate frequency (average number of transactions)
   - Moderate monetary value (average spending)
3. Low-Value: This last group of customers have the least attractive RFM values. Criteria include;
   - High recency (haven't made a purchase or transacted in a long time)
   - Lowest frequency
   - Lowest monetary value

[20]:
```
# Function to define categorization rules
# Categories to be used(High Value Customer, Average Customer, Low Value
 ↪Customer)

def categorize_customer(row):
    if row['Cluster'] == 3:
        return 'High Value'
```

```
    elif row['Cluster'] == 1:
        return 'Average'
    else:
        return 'Low Value'
```

[21]:
```
# Applyig the categorization function to each row of the rfm_table
rfm_table['CustomerCategory'] = rfm_table.apply(categorize_customer, axis = 1)
```

[22]:
```
rfm_table
```

[22]:
```
            Recency  Frequency   Monetary  Cluster CustomerCategory
CustomerID
12346           325          2  154367.20        3       High Value
12347             1          7    4310.00        1          Average
12348            74          4    1437.24        1          Average
12349            18          1    1457.55        1          Average
12350           309          1     294.40        2        Low Value
…               …          …         …        …              …
18280           277          1     180.60        2        Low Value
18281           180          1      80.82        2        Low Value
18282             7          3     179.50        1          Average
18283             3         16    2039.58        1          Average
18287            42          3    1837.28        1          Average

[4363 rows x 5 columns]
```

[23]:
```
category_count = rfm_table['CustomerCategory'].value_counts()
print(category_count)
```

```
CustomerCategory
Average       3239
Low Value     1108
High Value      16
Name: count, dtype: int64
```

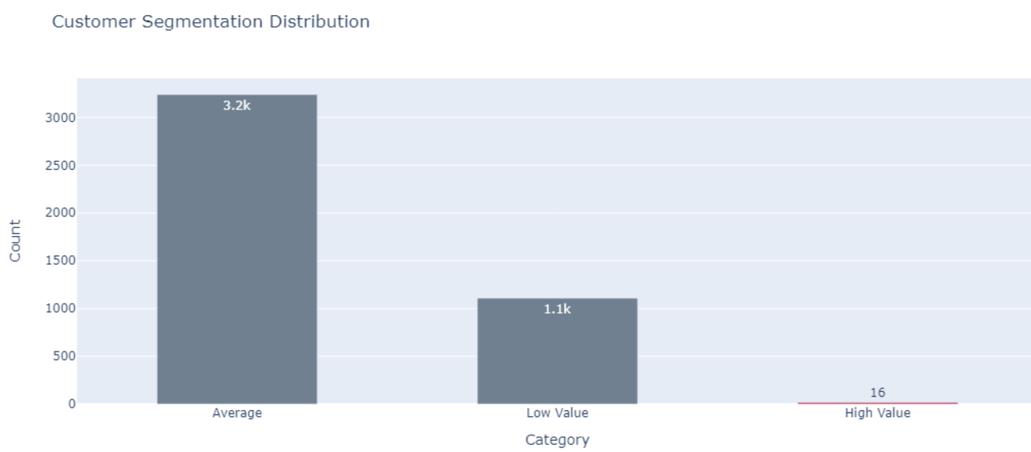
From the above output; - Average value customers have the most representation among the three clusters with a total of **3239**. This is followed by low value category of customers with **1108** and then high value customers with just **16**.

0.1.9 5. Visualization of Customer Segments using RFM Values

Bar charts, scatter plots, pie charts and a few other charts will be used to visualize relationship between the RFM values, and the data as a whole. We will be using both the clean data `online_retail_cl.csv` stored in the `df` variable and the `rfm_table` containing our RFM values and customer categorization.

5.1 Plot for the distribution of Customer Segments

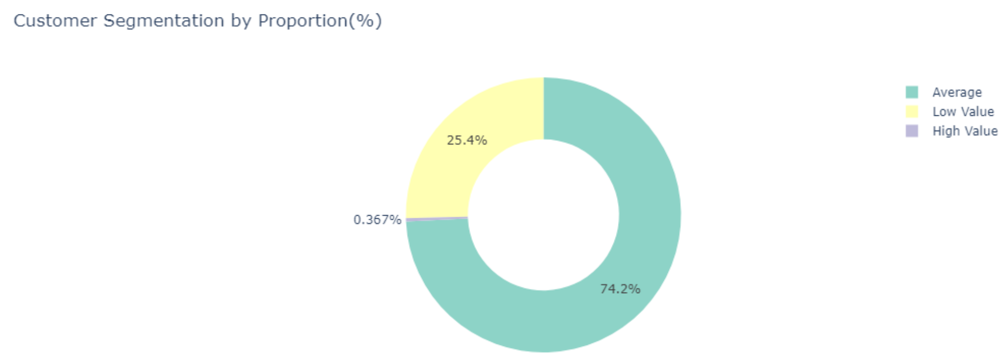
[24]:
```python
# Get the value counts of customer categories or segments
category_counts = rfm_table['CustomerCategory'].value_counts().reset_index()
category_counts.columns = ['CustomerCategory', 'Count']

# Creating a bar plot using plotly
fig = px.bar(category_counts, x='CustomerCategory', y='Count',
             labels={'CustomerCategory':'Category', 'Count':'Count'},
             title='Customer Segmentation Distribution',
            text_auto = '0.2s',
            hover_data = {'CustomerCategory'},)

# Customizing the color of high value category
colors = ['slategrey']*3
colors[2] = 'crimson'

fig.update_traces(marker_color=colors)

fig.update_layout(hovermode = 'x')
fig.update_layout(bargap=0.5)
fig.update_layout(width = 700, height = 500)
fig.show()
```



5.1.1 Customer Segment Distribution by Proportion (%)

[25]:
```python
# A pie chart will be employed in representing the proportions of each customer␣
↪segment

segment_prop = rfm_table['CustomerCategory'].value_counts().reset_index()
segment_prop_columns  = ['CustomerCategory', 'count']
```

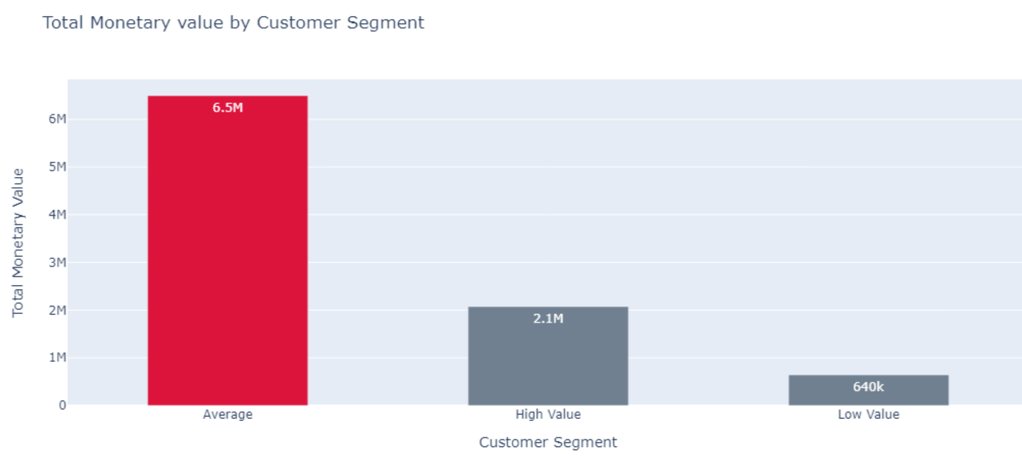
```
fig = px.pie(segment_prop, names = 'CustomerCategory', values = 'count',
            title = 'Customer Segmentation by Proportion(%)',
             hole = 0.55,
            color_discrete_sequence = px.colors.qualitative.Set3)

fig.update_layout(width = 650, height = 450)
fig.show()
```



5.2 Plotting Monetary values realized from each customer segment

[26]:
```
# Calculating the monetary value for each customer segment
monetary_total = rfm_table.groupby('CustomerCategory')['Monetary'].sum().
 ↪reset_index()

fig = px.bar(monetary_total, x='CustomerCategory', y='Monetary',
            title='Total Monetary value by Customer Segment',
            labels={'CustomerCategory':'Customer Segment', 'Monetary':'Total␣
 ↪Monetary Value'},
            #hover_data = {'CustomerCategory'},
            hover_data={'Monetary': ':,.2f'},
            text_auto = '0.2s')

colors = ['slategrey']*3
colors[0] = 'crimson'

fig.update_traces(marker_color=colors)

fig.update_layout(hovermode = 'x')
fig.update_layout(bargap=0.5)
fig.update_layout(width = 700, height = 500)
```

```
#fig.update_layout(xaxis_title='Customer Segment', yaxis_title='Monetary Value')
fig.show()
```



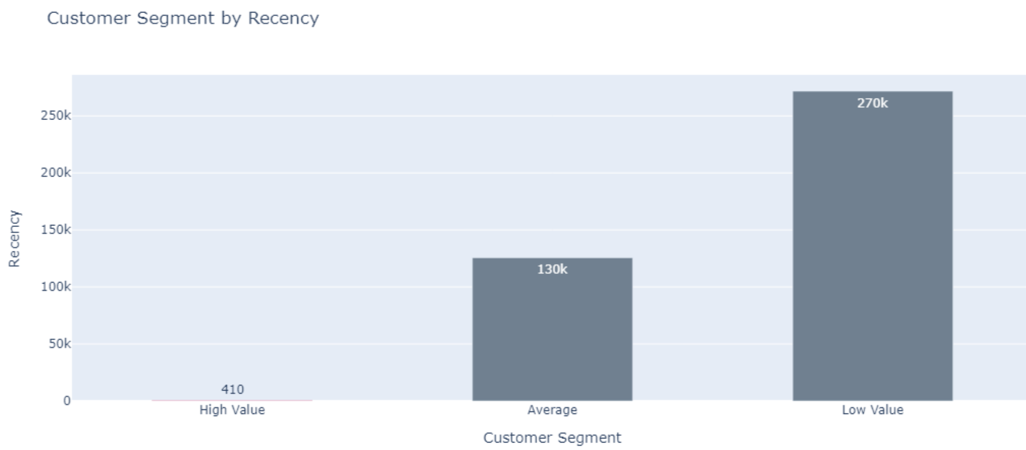
5.3 Plotting Recency values for each customer segment

[27]:
```
segment_recency_total = rfm_table.groupby('CustomerCategory')['Recency'].sum().
 ↪sort_values()
print(segment_recency_total)
```

```
CustomerCategory
High Value        411
Average        125624
Low Value      271673
Name: Recency, dtype: int64
```

[28]:
```
recency_total = segment_recency_total.reset_index()

fig = px.bar(recency_total, x = 'CustomerCategory', y = 'Recency',
            title = 'Customer Segment by Recency',
            labels= {'CustomerCategory':'Customer Segment', 'Recency':
 ↪'Recency'},
            category_orders = {'CustomerCategory':␣
 ↪sorted(rfm_table['CustomerCategory'].unique(),
                                                  key=lambda x :␣
 ↪rfm_table[rfm_table['CustomerCategory'] == x]['Recency'].mean())},
            hover_data = {'Recency' : ':,.0f'},
             text_auto = '0.2s')
colors = ['slategrey'] * 3
colors[0] = 'crimson'
```

```
fig.update_traces(marker_color = colors)
fig.update_layout(hovermode = 'x')
fig.update_layout(bargap = 0.5)
fig.update_layout(width = 700, height = 500)

fig.show()
```



5.4 Plotting the Frequency of each customer segment

[29]:

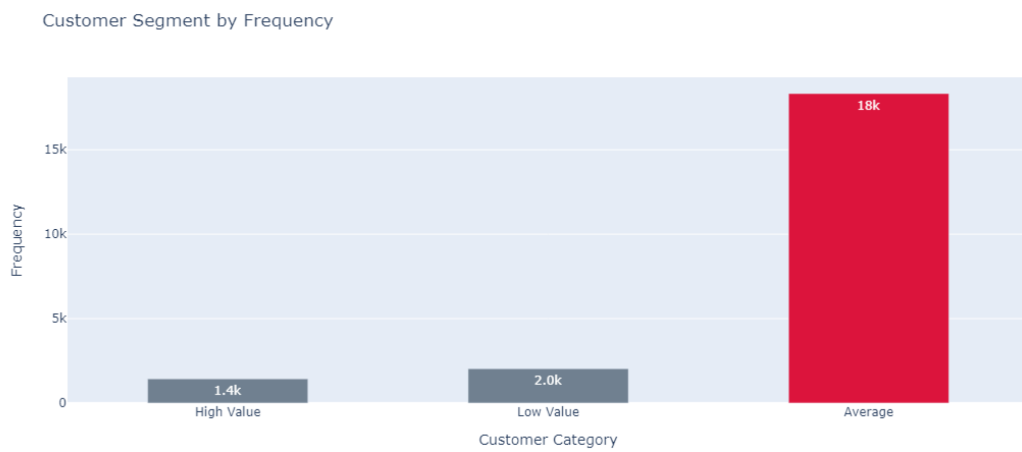
```
segment_frequency_total = rfm_table.groupby('CustomerCategory')['Frequency'].
  ↪sum().sort_values()
print(segment_frequency_total)
```

```
CustomerCategory
High Value      1433
Low Value       2030
Average        18325
Name: Frequency, dtype: int64
```

[30]:

```
frequency_total = segment_frequency_total.reset_index()
fig = px.bar(frequency_total, x = 'CustomerCategory', y = 'Frequency',
            title = 'Customer Segment by Frequency',
            labels = {'CustomerCategory':'Customer Category', 'Frequency':
  ↪'Frequency'},
            hover_data ={'Frequency':':,.0f'},
            text_auto = '0.2s')

colors = ['slategrey'] * 3
colors[2] = 'crimson'

fig.update_traces(marker_color = colors)
```

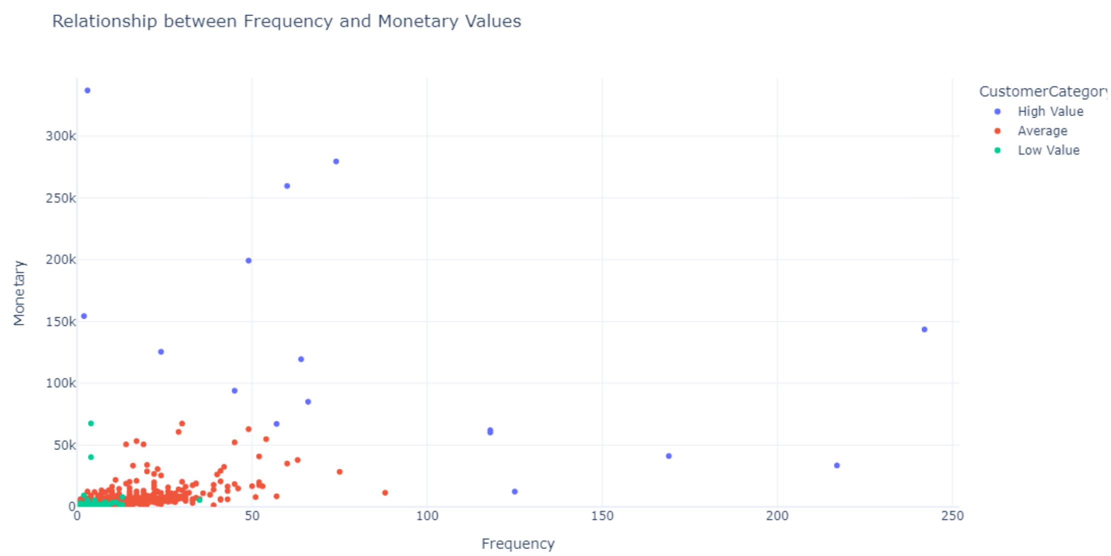
```
fig.update_layout(bargap = 0.5)
fig.update_layout(hovermode = 'x')
fig.update_layout(width = 700, height = 500)

fig.show()
```



5.5 Plot to visualize Relationship between RFM Metrics / Values For this task a scatter plot will be used to visualize the relationship between the two RFm values. This will help identify patterns across the different customer segments with respect to these two values.

5.5.1 Plot to visualize Relationship between Frequency and Monetary values

[31]:
```
# Creatig a scatter plot using Plotly
#customer_id = rfm_table[rfm_table['CustomerID']].unique()

fig = px.scatter(rfm_table, x = 'Frequency', y = 'Monetary',
                title = 'Relationship between Frequency and Monetary Values',
                labels = {'Frequency':'Frequency', 'Monetary':'Monetary'},
                 range_x = [0, rfm_table['Frequency'].max() + 10], # set x-axis␣
 ↪range starting from zero
                 range_y = [0, rfm_table['Monetary'].max() + 10000], # set␣
 ↪y-axis range to start from zero
                color = 'CustomerCategory', # color points based on customer␣
 ↪category
                template = 'plotly_white',
                hover_data = {'Monetary':':,.2f'},
                hover_name = 'Cluster')

fig.update_layout(height = 600)
fig.show()
```



From the scatter plot above, the following observations were made with respect to the relationship between `Frequency` which is the number of times a customer visited the online retail store and performed a transaction and `Monetary` which represents the total monetary value of each customer's trasanction. 1. `High Value` customers had the least representation on the plot with the lowest number of data points. The frequency values for this customer segment ranged from a frequency of **2** to **242**. This customer segment also recorded the highest monetary value among the three segments, with a monetary value of approximately **336,942K** been the highest and a value of approximately **12,365K** been the lowest in this segment.

2. From the plot, `Average` customers had the most data points. It is observed that the highest concentration of these data points fall between a frequency of **0** and **50**, and a monetary value of between **0** and approximately **33,923K**. For the segment, the highest frequency recorded is **88** and the highest monetary value is **67,387K**.

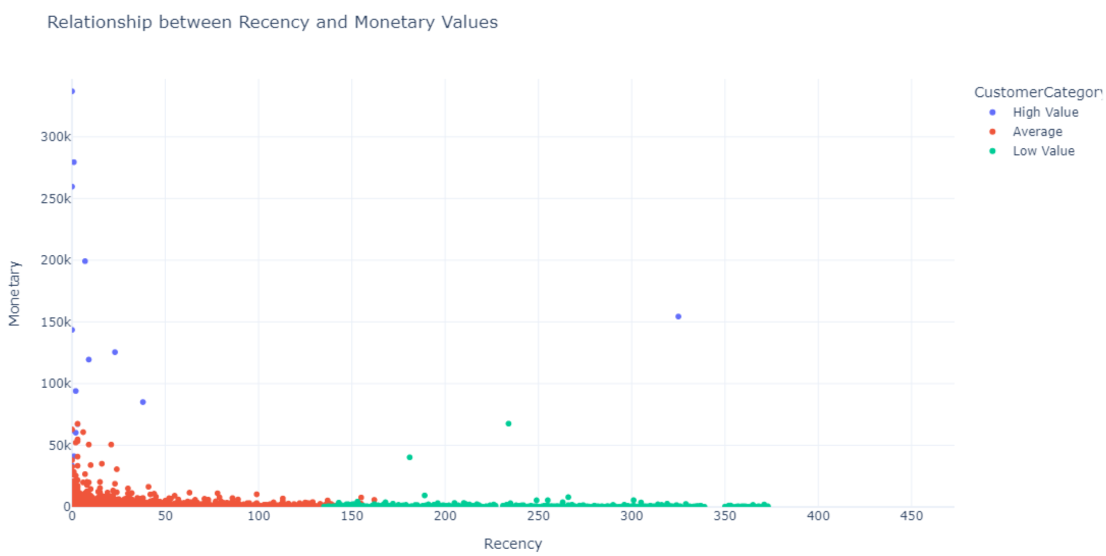
3. From the scatter plot, it is observed that most of the data points for `Low Value` customers fall between a frequency value of **1** and **13**, and monetary value falling between **1,693.88** and **9,344.14**. For this segment, the highest frequency value recorded is **35**, with the hgihest monetary value of **67,532K**.

We can conclude from the representation of data points on the scatter plot with respect to the three customer segment, that: - A higher frequency (Number of visits) does not translate into a corresponding higher monetary value and vice versa.

5.5.2 Plot to visualize Relationship between Recency and Monetary values

[32]:
```
fig = px.scatter(rfm_table, x = 'Recency', y = 'Monetary',
                 title = 'Relationship between Recency and Monetary Values',
                 labels = {'Recency':'Recency', 'Monetary':'Monetary'},
```

```
                range_x = [0, rfm_table['Recency'].max() + 100], # set x-axis␣
↪range starting from zero
                range_y = [0, rfm_table['Monetary'].max() + 10000], # set␣
↪y-axis range to start from zero
                color = 'CustomerCategory', # color points based on customer␣
↪category
                template = 'plotly_white',
                hover_data={'Monetary':':,.0f'},
                hover_name = 'Cluster')

fig.update_layout(height = 600)
fig.show()
```



Insight: From the above scatter plot above, the following observation was made of the relationship between `Recency` which represents how current or recent a customer's visit to the online retail store is and the `Monetary` value realized from their visit.

1. `High Value` customers had some of the most recent visits to the online retail store. The recency values for this segment ranged from **0** days to **325** days since a customer of this segment last visited the online retail store. In terms of monetary value of these visits, we observed very high monetary values realized for visits with a recency of zero. This recency value of zero represents visits that are less than 24 hours, meaning most high value segment customers either visited the online store at or within the same time range. From the visualization we can say the most monetary gain from high value customers was realized between **0** and **38** recency values. Only one had a recency value of **325**.

2. For `Average` customers, their recency values ranged from **0** days(same day or within 24 hours) to **162** days since a customer in this segment or category last visited the online retail store. In

terms of monetary value for the segment or category,we realised a monetray value of **67,387K** for a recency of **3** days. This was the highest in this segment and with monetary values as low as **298**. Form the scatter plot, we can say that the most monetary gain for this customer segment was realised between **0** and **24** recency values.

3. `Low value` customers had the highest recency values comparatively among the three customer segment. The recency values for this segment ranged from **135** days to **373** days. In terms of monetary value for this segment, we observed values as low as **168** with a recency of **339** days since the last visit. Although this segment recorded very low values overall, we observed a couple of low value data points that had a monetary value as high as **67,533** and a recorded recency value of **234**.

[33]:
```
# Filtering out data on high value customers with a recency value of 0
rfm_filter = rfm_table[(rfm_table['Recency'] == 0) &␣
 ↪(rfm_table['CustomerCategory'] == 'High Value')]
rfm_filter.sort_values(by = 'Monetary', ascending = False)
```

[33]:

CustomerID	Recency	Frequency	Monetary	Cluster	CustomerCategory
16446	0	3	336942.10	3	High Value
18102	0	60	259657.30	3	High Value
14911	0	242	143555.42	3	High Value
15311	0	118	61981.31	3	High Value
12748	0	217	33481.57	3	High Value
14606	0	125	12365.70	3	High Value

From the above table, we see that customer with ID **16446** recorded the least frequency value of **3** but also recorded the highest monetary value of **336,942K**.

2. Majority of the customers of the online retail store fall in the `Average` customer segment. This segment recorded a recency value ranging from **0** to **162**.

[34]:
```
# Filtering out data on average customers based on their monetary value and␣
 ↪recency
rfm_filter = rfm_table[rfm_table['CustomerCategory'] == 'Average']
rfm_filter.sort_values(by = 'Monetary', ascending = False)
```

[34]:

CustomerID	Recency	Frequency	Monetary	Cluster	CustomerCategory
16684	3	30	67387.00	1	Average
17949	0	49	62845.22	1	Average
15769	6	29	60681.72	1	Average
15061	3	54	54817.94	1	Average
14096	3	17	53258.43	1	Average
…	…	…	…	…	…
13307	119	1	15.00	1	Average
14792	63	2	12.40	1	Average
16454	63	1	5.90	1	Average
16428	80	1	2.95	1	Average

```
13256              13           1       0.00         1           Average

[3239 rows x 5 columns]
```

Insight: From the table above, the highest monetary value realized from this segment is **67,387.00K** with a recency value of **3**, meaning customer with ID **16684** visited the online store very recently and during this time this customer visited the store **30** times. We can also see that customer with ID **13256** last visit to the online store was 13 days ago from our data and visited the site only once and performed no transaction of any kind accounting for a monetary value of **0.00**.

3. `Low Value` customers account for the second highest number of customers in our data. Customers within this segment recorded a recency value of between **135** to **373**. In terms of monetary value, a high of **67,532.70K** was realized to as low as **1.25**.

[35]:
```
# Filtering out data on low value customers based on their monetary value and␣
↪recency
rfm_filter = rfm_table[rfm_table['CustomerCategory'] == 'Low Value']
rfm_filter.sort_values(by = 'Monetary', ascending = False)
```

[35]:
```
            Recency  Frequency  Monetary  Cluster CustomerCategory
CustomerID
15749           234          4  67532.70        2        Low Value
15098           181          4  40213.50        2        Low Value
12590           189          2   9344.14        2        Low Value
13093           266         13   7923.47        2        Low Value
17850           301         35   5478.94        2        Low Value
…                 …         …         …        …                …
12605           364          1      7.50        2        Low Value
16738           297          1      3.75        2        Low Value
12943           300          1      3.75        2        Low Value
14679           371          1      2.55        2        Low Value
16995           371          1      1.25        2        Low Value

[1108 rows x 5 columns]
```

Insight: The table above shows customer with ID **15749** visited the online store **234** days ago since his or her last visit and in that time visited the store only **4** times but still recorded the highest monetary value of **67,532.70** in this segment.